IMPRIMINDO NA TELA EM ASSEMBLY

Prof. Bruno Silvestre

A seguir serão apresentados modelos de como imprimir uma string, string sem formatação, inteiros, hexadecimal e caractere.

O printf() verifica o registrador %rax para saber quantos valores de ponto-flutuante foram passados como argumento. Como em todos os modelos abaixo não é passado nenhum ponto-flutuante como parâmetro, %rax será sempre zero.

A string de formatação é colocada como uma constante na seção ".rodata" por questão de segurança e evitar que a string seja modificada.

Informações sobre formatação do *printf*():
http://www.cplusplus.com/reference/cstdio/printf/

1) Imprimindo uma string sem formatação

- A string não contém nenhuma marcação de formatação “%”
- O ponteiro da string de formatação deve ser colocado em %rdi
- Zerar o registrador %rax

Exemplo:

```
.section .rodata

fmt: .string "Hello world\n"

.text
main:

   # Código omitido, somente trecho do printf exibido.

    movq $fmt, %rdi
    movl $0, %eax      # Zerar o registrador.
    call printf
```

2) Imprimindo um inteiro de 8-bits ou 16-bits

- Deve ser usado “%d” (com sinal) ou “%u” (sem sinal) na string
- O ponteiro da string de formatação deve ser colocado em %rdi
- Os valores inteiros (8-bits ou 16-bits) devem ser estendidos para 32-bits e colocados nos respectivos registradores de parâmetros inteiros (%esi, %edx, %ecx, %r8d, %r9d)
- Zerar o registrador %rax

Exemplo:

```
.section .rodata

fmt: .string "Value: %d\n"

.data

.align 2
i: .short 20

.text
main:

    # Código omitido, somente trecho do printf exibido.

    movq   $fmt, %rdi
    movswl    i, %esi    # Estende de 16 para 32-bits.
    movl   $0, %eax      # Zerar o registrador.
    call   printf
```

3) Imprimindo um inteiro 32-bits

- Deve ser usado “%d” (com sinal) ou “%u” (sem sinal) na string
- O ponteiro da string de formatação deve ser colocado em %rdi
- Os valores inteiros 32-bits devem ser colocados nos respectivos registradores de parâmetros inteiros (%esi, %edx, %ecx, %r8d, %r9d)
- Zerar o registrador %rax

Exemplo:

```
.section .rodata

fmt: .string "Valuor %d maior que %d\n"

.data

.align 4
i: .int  10
j: .int -20

.text
main:


    # Código omitido, somente trecho do printf exibido.

    movq $fmt, %rdi
    movl    i, %esi     # primeiro “%d” → i
    movl    j, %edx     # segundo  “%d” → j
    movl $0, %eax       # Zerar o registrador.
    call printf
```

4) Imprimindo um inteiro 64-bits

- Deve ser usado “%ld” (com sinal) ou “%lu” (sem sinal) na string
- O ponteiro da string de formatação deve ser colocado em %rdi
- Os valores inteiros 64-bits devem ser colocados nos respectivos registradores de parâmetros inteiros (%rsi, %rdx, %rcx, %r8, %r9)
- Zerar o registrador %rax

Exemplo:

```
.section .rodata

fmt: .string "Value: %ld\n"

.text
main:

    # Código omitido, somente trecho do printf exibido.

    movq $fmt, %rdi
    movq  %12, %rsi    # Pode-se colocar um valor constante
                       # ou valor de variável em %rsi.
    movl $0,   %eax    # Zerar o registrador.
    call printf
```

5) Imprimindo um endereço de memória (Ponteiro)

- Deve ser usado “%p” na string
- O ponteiro da string de formatação deve ser colocado em %rdi
- Os endereços de memória devem ser colocados nos respectivos registradores 64-bits de parâmetros inteiros (%rsi, %rdx, %rcx, %r8, %r9)
- Zerar o registrador %rax

Exemplo:

```
.section .rodata

fmt: .string "Endereco de x: %p\n"

.data

.align 4
x:   .int 1024

.text
main:

    # Código omitido, somente trecho do printf exibido.

    movq $fmt, %rdi
    movq $x,   %rsi   # Pode-se usar leaq para obter endereços de memória.
    movl $0,   %eax   # Zerar o registrador.
    call printf
```

6) Imprimindo um valor em hexadecimal (32-bits)

- Deve ser usado “%X” na string
- O ponteiro da string de formatação deve ser colocado em %rdi
- Os valores inteiro 32-bits (sem sinal) devem ser colocados nos respectivos registradores de parâmetros inteiros (%esi, %edx, %ecx, %r8d, %r9d)
- Zerar o registrador %rax

Exemplo:

```
.section .rodata

fmt: .string "Valor: %X\n"

.text
main:

    # Código omitido, somente trecho do printf exibido.

    movq $fmt, %rdi
    movl $256, %esi   # Valor para "%X" é um inteiro 32-bits.
    movl $0, %eax     # Zerar o registrador.
    call printf
```

7) Imprimindo um caractere

- Deve ser usado “%c” na string
- O ponteiro da string de formatação deve ser colocado em %rdi
- Os caracteres devem ser estendidos para inteiro 32-bits (com sinal) e devem ser colocados nos respectivos registradores de parâmetros inteiros (%esi, %edx, %ecx, %r8d, %r9d)
- Zerar o registrador %rax

Exemplo:

```
.section .rodata

fmt: .string "Letra: %c\n"

.text
main:

    # Código omitido, somente trecho do printf exibido.

    movq $fmt, %rdi
    movl $65,  %esi    # Valor para "%c" é um inteiro 32-bits.
    movl $0,   %eax    # Zerar o registrador.
    call printf
```

8) Imprimindo uma string

- Deve ser usado “%s” na string
- O ponteiro da string de formatação deve ser colocado em %rdi
- Os endereços das strings devem ser colocados nos respectivos registradores 64-bits de parâmetros inteiros (%rsi, %rdx, %rcx, %r8, %r9)
- Zerar o registrador %rax

Exemplo:

```
.section .rodata

fmt: .string "Hello %s\n"



# Considerando 'str' como variável. Se for constante,
# poderia mover para seção anterior de read-only data.
# Isso não afeta o código abaixo.

.data

str: .string "world"

.text
main:

    # Código omitido, somente trecho do printf exibido.

    movq $fmt, %rdi
    movq $str, %rsi
    movl $0,   %eax      # Zerar o registrador.
    call printf
```